Report

***“Limit Power Consumption on Home Appliances using Machine Learning”***

**By,**

Sudhanva Narayana,

Intern, Machine Learning Engineer,

**Phone:** +91 9632350260

**Email:** [nsudhanva@gmail.com](mailto:nsudhanva@gmail.com)

**Faststream Technologies,**

6, 2nd Main Rd, Arekere Off Bannerghatta Road, Bengaluru, KA, 560076

**Phone:** +91 80–4161–3204/3205

**Email:** [info@faststreamtech.com](mailto:info@faststreamtech.com)

|  |  |
| --- | --- |
| **Table of Contents** | **Page** |
| **Executive Summary** | **No** |
| 1. **Introduction**   *1.1. Background*  *1.2. Home Automation*  *1.3. Machine Learning*  *1.4. Project Objectives*  *1.5. Dependencies and Tools*  *1.6. Limitations of the project* | **1**  2  3  4  5  6  7 |
| 1. **Preparing Dataset**   *2.1. Generating Data*  *2.2. Dataset description* | **8**  10  11 |
| 1. **Project Methodologies**   *3.1. Visualization*  *3.2. Applying Machine Learning*  *3.3. Prediction*  3.4. Custom Algorithm | **12**  13  15  16  16 |
| 1. **Results**   *4.1. Multiple Linear Regression*  *4.2. Random Forest Regression*  *4.3. Decision Tree Regression*  *4.4. Conclusion* | **20**  21  22  23  24 |
| **References** | **25­­­­­** |
| **Appendix** | **26** |

**1. INTRODUCTION**

Power (electricity) optimization will be a crucial task in the coming years as there will be a limited supply. While we run out of electricity, it will affect both domestic and industrial applications. This report demonstrates techniques and strategies to optimize power consumption when there is a limited supply, using Machine Learning to predict the amount of electricity consumed at a given hour for a given device specifically for home appliances. It also notifies the user of excess power consumption and also suggests measures to save power. It also considers factors like weather, time of the day and type of the device and the device priority to make the final decision. Machine Learning algorithms such as Multiple Linear Regression, Decision Tree Regression, Random Forest Regression are used to predict the values. We present the result of each algorithm by showing in how much each algorithm is better (or worse) compared to the rest of the algorithms **[1]**.

***1.1. Background***

Faststream Technologies is a vanguard of technology solutions, specializing in Product & System Engineering, IoT, Big Data, Security, and Application Development with a global footprint across North America, EMEA, and APAC. With over 200+ clients, Faststream Technologies enables Digital Transformation for enterprises by delivering a flawless customer experience, business competence, and deep insights through an integrated set of disruptive technologies and expertise. We are passionate about delivering well-organized, inventive and world-class hardware and software solutions, with a focus on Healthcare, Aerospace, Semiconductors, Automotive, Consumer Electronics, Home Automation, Telecommunications, Security, Retail, and E-Commerce.

Faststream Technologies works at the juncture of business and technology, assisting clients with advancing their product and business performance through sustainable information technology solutions. Faststream Technologies drives innovation to help clients advance their product design, business processes, and application development. Our engineering team’s deep expertise in transforming design specs into marketable hardware products — through ASIC design services that include RTL design, design verification and physical design for digital and analogue/mixed-signal semiconductors — is a key differentiator to our suite of application development capabilities.

For today’s challenges like embedded processor SoC specifications, Faststream Technologies delivers all of the required firmware/embedded software, positioning us as the turnkey ‘concept-to-product’ design company. The team is led by a group of focused senior executives and Technologists who complement each other with significant industry experience in building turnkey solutions. Many of our technologists have multiple patents to their credit in the areas of Analog/Mixed-Signal Design, IoT and embedded systems.

***1.2. Home Automation***

Home automation or demotics is building automation for a home, called a smart home or smart house. A home automation system will control lighting, climate, entertainment systems, and appliances. It may also include home security such as access control and alarm systems. When connected to the Internet, home devices are an important constituent of the Internet of Things **[2]**. A home automation system typically connects controlled devices to a central hub or "gateway". The user interface for control of the system uses either wall-mounted terminals, tablet or desktop computers, a mobile phone application, or a Web interface, that may also be accessible off-site through the Internet. While there are many competing vendors, there are very few worldwide accepted industry standards and the smart home space is heavily fragmented. Manufacturers often prevent independent implementations by withholding documentation and by litigation. The home automation market was worth US$5.77 billion in 2013, predicted to reach a market value of US$12.81 billion by the year 2020.

***1.3. Machine Learning***

Machine Learning is the science (and art) of programming computers so they can learn from data. For example, your spam filter is a Machine Learning program that can learn to flag spam given examples of spam emails (e.g., flagged by users) and examples of regular (non-spam) emails. The examples that the system uses to learn are called the training set. Each training example is called a training instance (or sample). In this case, the task T is to flag spam for new emails, the experience E is the training data, and the performance measure P needs to be defined; for example, you can use the ratio of correctly classified emails. This particular performance measure is called accuracy and it is often used in classification tasks **[3]**.

***Supervised Learning***

Machine Learning systems can be classified according to the amount and type of supervision they get during training. There are four major categories: supervised learning, unsupervised learning, semi-supervised learning, and Reinforcement Learning. In supervised learning, the training data you feed to the algorithm includes the desired solutions, called labels. A typical supervised learning task is classification. The spam filter is a good example of this: it is trained with many examples of emails along with their class (spam or ham), and it must learn how to classify new emails.

***Unsupervised learning***

In unsupervised learning, as you might guess, the training data is ranged. The system tries to learn without a teacher.

* **Clustering** 
  + k-Means
  + Hierarchical Cluster Analysis (HCA)
  + Expectation Maximization
* **Visualization and dimensionality reduction** 
  + Principal Component Analysis (PCA)
  + Locally-Linear Embedding (LLE)
  + t-distributed Stochastic Neighbour Embedding (t-SNE)

***1.4. Project Objectives***

We use multiple regression techniques to predict power consumption based on previously given data. After the values are predicted, a custom algorithm is used to limit power consumption by identifying devices that are consuming more power, by type of weather, by rooms and notify the user of its excess power consumption. It also considers factors like a number of people in the room and time they have stayed in that room. In this way, the total power consumption of devices is reduced per month. However, the power consumption per day is not considered limited. Machine Learning is used only to predict the power consumption at that point in time and not to predict or understand the behaviour of people. As there is no real data, we used a random number generation and time series generator to generate a dataset of certain attributes on which we used to train our models. We later used another generated dataset to predict values from the trained model using regression **[4]**.

Regression is a set of statistical processes for estimation of the relationship among data points and variables. It predicts the conditional expectation of the dependent variable when given the independent variables, that is the average of both dependent and independent variables. A function of the independent variables is estimated and the values are predicted using probability distribution.

**List of regression algorithms used:**

* **Multiple Linear Regression**
  + Multiple linear regression (MLR) is a multivariate statistical technique for examining the linear correlations between two or more independent variables (IVs) and a single dependent variable (DV).
* The population regression line for *p* explanatory variables *x*1, *x*2, ... , *x*p is defined to be http://www.stat.yale.edu/Courses/1997-98/101/mu2.gify = http://www.stat.yale.edu/Courses/1997-98/101/beta.gif0 + http://www.stat.yale.edu/Courses/1997-98/101/beta.gif1*x*1 + http://www.stat.yale.edu/Courses/1997-98/101/beta.gif2*x*2 + ... + http://www.stat.yale.edu/Courses/1997-98/101/beta.gifp*x*p. This line describes how the mean response http://www.stat.yale.edu/Courses/1997-98/101/mu2.gify changes with the explanatory variables. The observed values for *y* vary about their means http://www.stat.yale.edu/Courses/1997-98/101/mu2.gify and are assumed to have the same standard deviation http://www.stat.yale.edu/Courses/1997-98/101/sigma2.gif. The fitted values *b0*, *b1*, ..., *bp* estimate the parameters http://www.stat.yale.edu/Courses/1997-98/101/beta.gif0, http://www.stat.yale.edu/Courses/1997-98/101/beta.gif1, ..., http://www.stat.yale.edu/Courses/1997-98/101/beta.gifp of the population regression line **[6]**
* Since the observed values for *y* vary about their means http://www.stat.yale.edu/Courses/1997-98/101/mu2.gify, the multiple regression model includes a term for this variation. In words, the model is expressed as DATA = FIT + RESIDUAL, where the "FIT" term represents the expression http://www.stat.yale.edu/Courses/1997-98/101/beta.gif0 + http://www.stat.yale.edu/Courses/1997-98/101/beta.gif1*x*1 + http://www.stat.yale.edu/Courses/1997-98/101/beta.gif2*x*2 + ... http://www.stat.yale.edu/Courses/1997-98/101/beta.gifp*x*p. The "RESIDUAL" term represents the deviations of the observed values *y* from their means http://www.stat.yale.edu/Courses/1997-98/101/mu2.gify, which are normally distributed with mean 0 and variance http://www.stat.yale.edu/Courses/1997-98/101/sigma2.gif. The notation for the model deviations is http://www.stat.yale.edu/Courses/1997-98/101/eps.gif.
* **Formally, the model for multiple linear regression, given *n* observations, is   
  *y*i =**http://www.stat.yale.edu/Courses/1997-98/101/beta.gif**0 +**http://www.stat.yale.edu/Courses/1997-98/101/beta.gif**1*x*i1 +**http://www.stat.yale.edu/Courses/1997-98/101/beta.gif**2*x*i2 + ...**http://www.stat.yale.edu/Courses/1997-98/101/beta.gif**p*x*ip +**http://www.stat.yale.edu/Courses/1997-98/101/eps.gif**i for *i* = 1,2, ... *n*.**
* **Random Forest Regression**
  + Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.
* **Decision Tree Regression**
  + Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in statistics, data mining and machine learning.

*The result is interpreted and plotted to measure the performance of the mentioned algorithms above.*

***1.5. Dependencies and Tools***

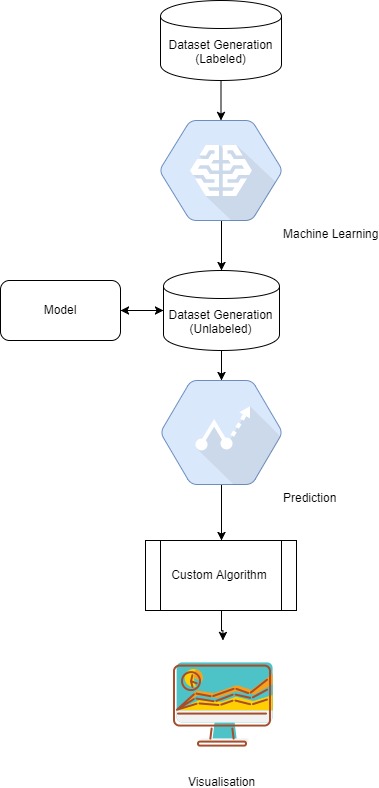
1. Python – a general-purpose interpreted, interactive, object-oriented, and high-level programming language
2. Anaconda – a free and open source distribution of the Python and R programming languages for data science and machine learning related applications, that aims to simplify package management and deployment
3. Numpy – the fundamental package for scientific computing with Python.
4. Scipy - a Python-based ecosystem of open-source software for mathematics, science, and engineering
5. Pandas - pandas is an open source, a BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
6. Matplotlib – a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms
7. Sci-kit learn – a free software machine learning library for the Python programming language.

***1.6. Limitations of the project***

The images are generated using numpy’s random number generator and mesh-grid technique which uses sine and cosine function to generate contour like matrices. However, these generated images are not real and should be used for experimental purposes only. The accuracy, methodology of the machine learning algorithms is true whereas the images are not. The project is also dependent on certain Python environments and related tools. It is independent of the development environment.

**2. PREPARING DATASET**

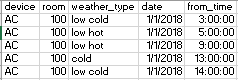
**Steps involved**

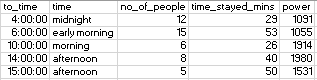
******

Data is checked for validity, accuracy, completeness and consistency. Data auditing is made to detect anomalies and contradictions. The detection and removal of anomalies are performed by a sequence of operations on the data known as the workflow. After executing the cleansing workflow, the results are inspected to verify correctness. Data that could not be corrected during execution of the workflow is manually corrected.

**2.1. Generating data**

As there is no real data, we are using a random number generator and time series generator to generate a dataset. The dataset we generated consists of multiple attributes such as device, room, weather type, date, from time, to time, time of day, a number of people and time stayed. We use the python’s popular numerical computation library ‘numpy’ and data structure ‘pandas’ to generate random numbers for the dataset and time series data. A sample dataset is shown below





**2.2. Dataset description**

Column ‘power’ is also generated along with it in terms of kilo watt hour.

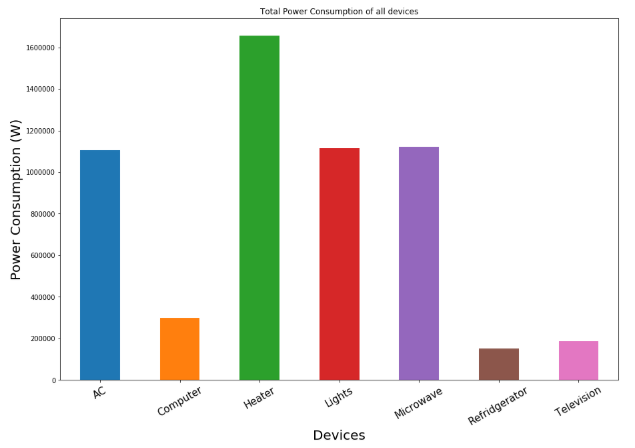
* Device: All home appliances
* Rooms: (any room number)
* Weather Types: cold, hot – low, medium, very
* Date – one month
* From/To - 1 hr frequency
* Time – morning, afternoon, evening, midnight
* Power consumption – Kilo Watt-hr

7 devices, 6 weather types, 1 month of date range, data of every device for every single date range are generated. Random values are generated for a number of people, time stayed in minutes and weather type, rooms, devices are randomly selected. Time of the day is selected as per the time series. We generated one dataset with power consumption and one without power consumption.

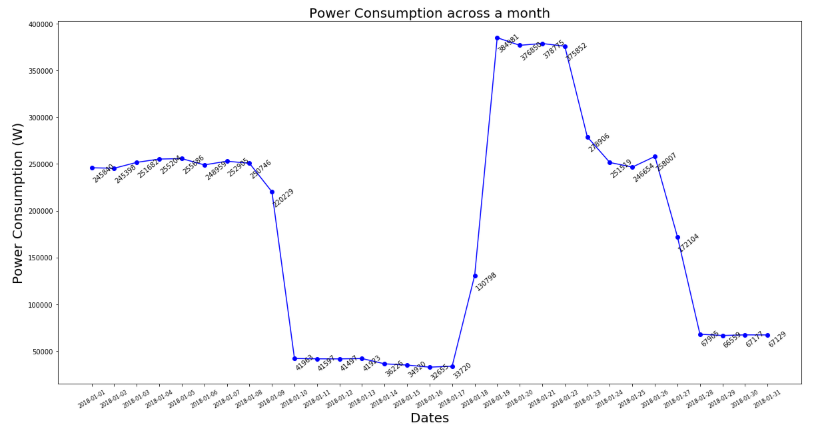
**3. PROJECT METHODOLOGIES**

Once the dataset is ready, it is loaded into our program. We use python’s ‘pandas’ library to load in data as a data frame. We visualise the dataset in terms of power consumption across all devices **(Figure 2)**. From the chart, we can see that AC and Heater consume more power than the rest of the devices. We also plot total power consumption every day and power consumption at every hour.

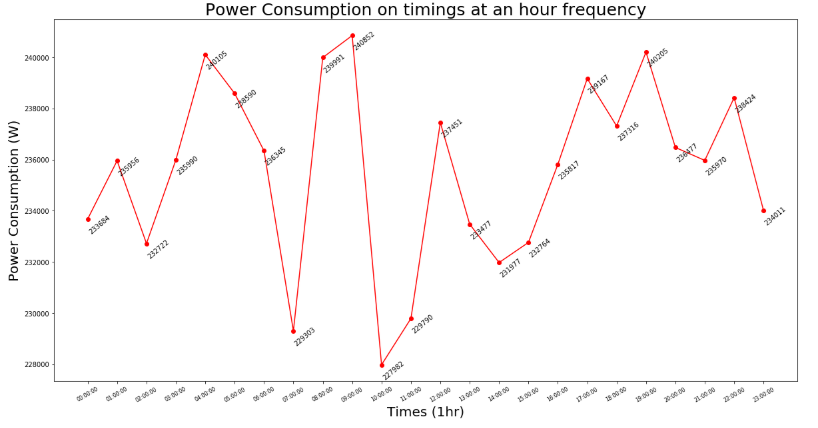
**3.1. Visualization**



***Figure 2 – Power consumption of all devices***

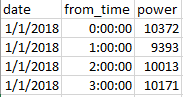


***Figure 3 – Power consumption (daily frequency)***



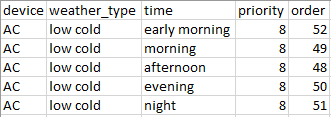
***Figure 4 – Power consumption (hourly frequency)***

A database consisting of maximum power consumption per day and total power consumption per month is created. A sample database is shown in **Figure 5.** This database has a date, time and maximum power consumption limit at that time.



***Figure 5 – Maximum power limit at that hour***

Another database containing devices and their priority in terms of time of the day and the type of weather is created. A sample database is shown in **Figure 6**.



***Figure 6 – Priority database of devices***

**3.2. Applying Machine Learning model to data**

1. The training set is determined. Here, 75% of the generated dataset is used as training data and the rest 25% is test data.
2. The training set represents the real-world use if the function. A set of input objects is gathered and corresponding outputs are also gathered.
3. The input representation of the learn function is determined and the accuracy of the learned function depends strongly on how the input object is represented. The input object is transformed into a feature vector, which contains a number of features that are descriptive of the object.
4. The structure of the learned function is determined (any Machine Learning algorithm can be used)
5. The design (model) is completed and I run on the gathered training set. Some of the supervised algorithms require the user to determine certain control parameters. These parameters are adjusted by optimising performance on the test set of the validation set.

**3.3. Prediction**

The next step involves applying machine learning to build our model. The data is split into a training set and test set and is evaluated based on the regression validation metrics. The algorithms specifically used are multiple linear regression, random forest regression and decision tree regression. Since the dataset contains multiple independent features and only one dependent feature, these algorithms are appropriate to be used. The model is trained on these algorithms separately and is saved.

Now, the unseen data is loaded, the saved model is loaded and is used to predict the dependent variable, that is the power consumption values. Below is a sample of the data including the power column, that is predicted by one the algorithms used above.

**3.4. Custom Algorithm**

Based on the previously created maximum power consumption database, a custom algorithm is used to give out a message and action to be performed. The message and action are given based on the priority database created. A new dataset is created with messages, actions and power saved is appended to the existing database. A sample message looks like this.

*“Moving 10 people from room 119 to room 105 saves 1188.0 of electricity, power consumption will reduce from 10507.0 to 9319.0”.* A sample action looks like this “*Turn off AC in room 119”.*

Code for the algorithm is given below:

# Creates priorities for devices

priorities = []

for index, row in sample\_df.iterrows():

priorities.append(df\_home\_priority[(df\_home\_priority['device'] == row['device'])

& (df\_home\_priority['weather\_type'] == row['weather\_type'])

& (df\_home\_priority['time'] == row['time'])].values[0][4])

# Keep looping till all rows are appended created and appended

while sample\_total\_power < sample\_after\_power:

no\_of\_people\_index = sample\_df.iloc[[-2]]['no\_of\_people'].index[0]

last\_one\_room = sample\_df.iloc[[-2]]['room'].values[0]

last\_no\_of\_people = sample\_df.iloc[[-1]]['no\_of\_people'].values[0]

last\_one\_no\_of\_people = sample\_df.iloc[[-2]]['no\_of\_people'].values[0]

last\_room = sample\_df.iloc[[-1]]['room'].values[0]

last\_device = sample\_df.iloc[[-1]]['device'].values[0]

sample\_df.at[no\_of\_people\_index, 'no\_of\_people'] = last\_no\_of\_people + last\_one\_no\_of\_people

sample\_df\_drop = sample\_df.drop(sample\_df.tail(1).index)

sample\_after\_power = sample\_total\_power - sample\_df\_drop['power'].sum()

# Gives messages to users

action = 'Turn off ' + last\_device + ' in room ' + str(last\_room)

if last\_no\_of\_people != 0:

message = 'Moving ' + str(last\_no\_of\_people) + ' people from room ' + str(last\_room) + ' to room ' + str(last\_one\_room) + ' saves ' + str(sample\_after\_power) + ' of' + ' electricity, ' + 'power consumption will reduce from ' + str(sample\_total\_power) + ' to ' + str(sample\_total\_power - sample\_after\_power)

else:

message = 'None'

# Appends decided messages

messages.append(message)

actions.append(action)

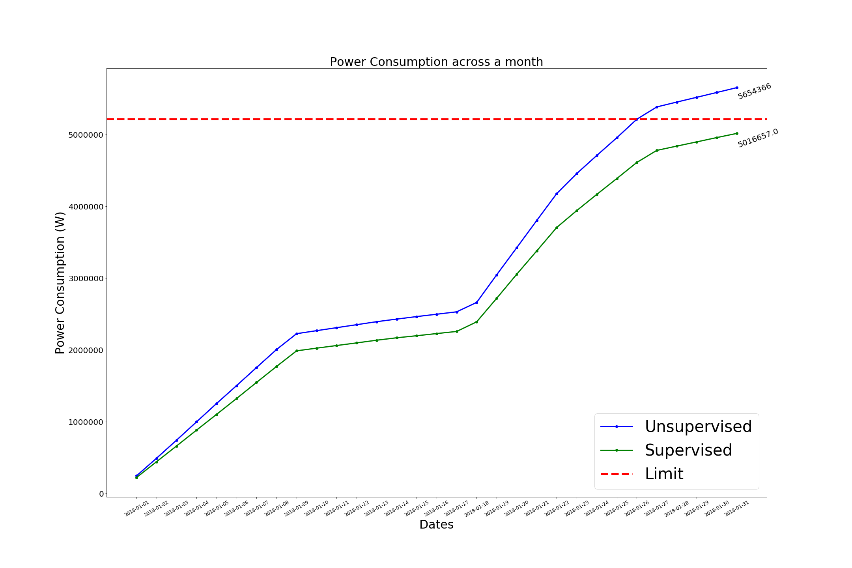
savings.append(sample\_after\_power)

The algorithm checks for priority of the devices then compares it with mean power consumption database of the model and the unseen data. If the mean power consumption is greater than the model, then the actions and messages are displayed. It also considers number of people and time they have stayed in the room as important factors. Finally, it calculates the power saved, of the devices.

**4. RESULTS**

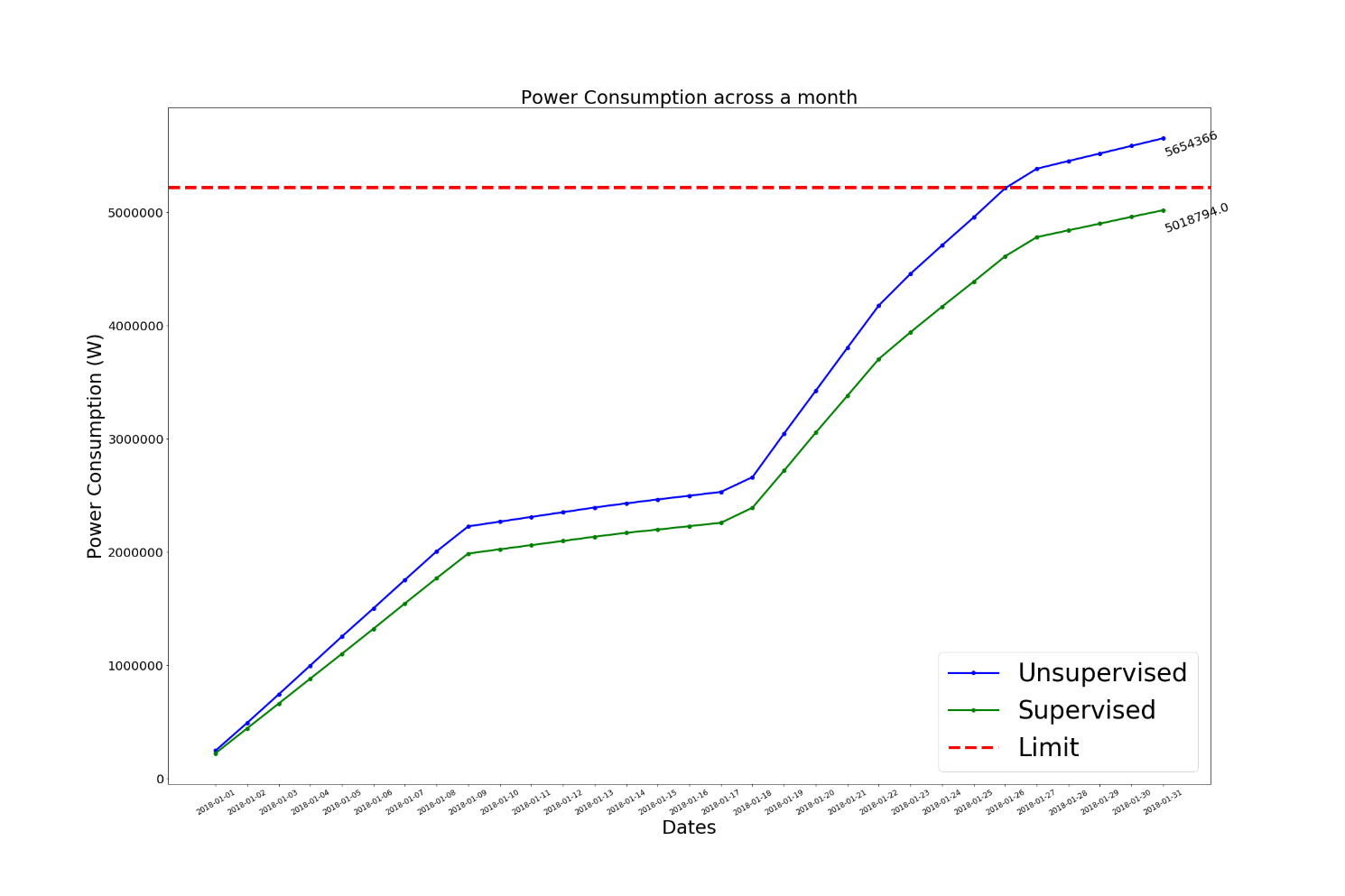
We take the mean power consumption per day for that hour versus the mean power consumption that was previously specified and then plots the outcomes. We do this for all the three machine learning algorithms.

**4.1. Multiple Linear Regression**



***Figure 7 – Multiple Linear Regression***

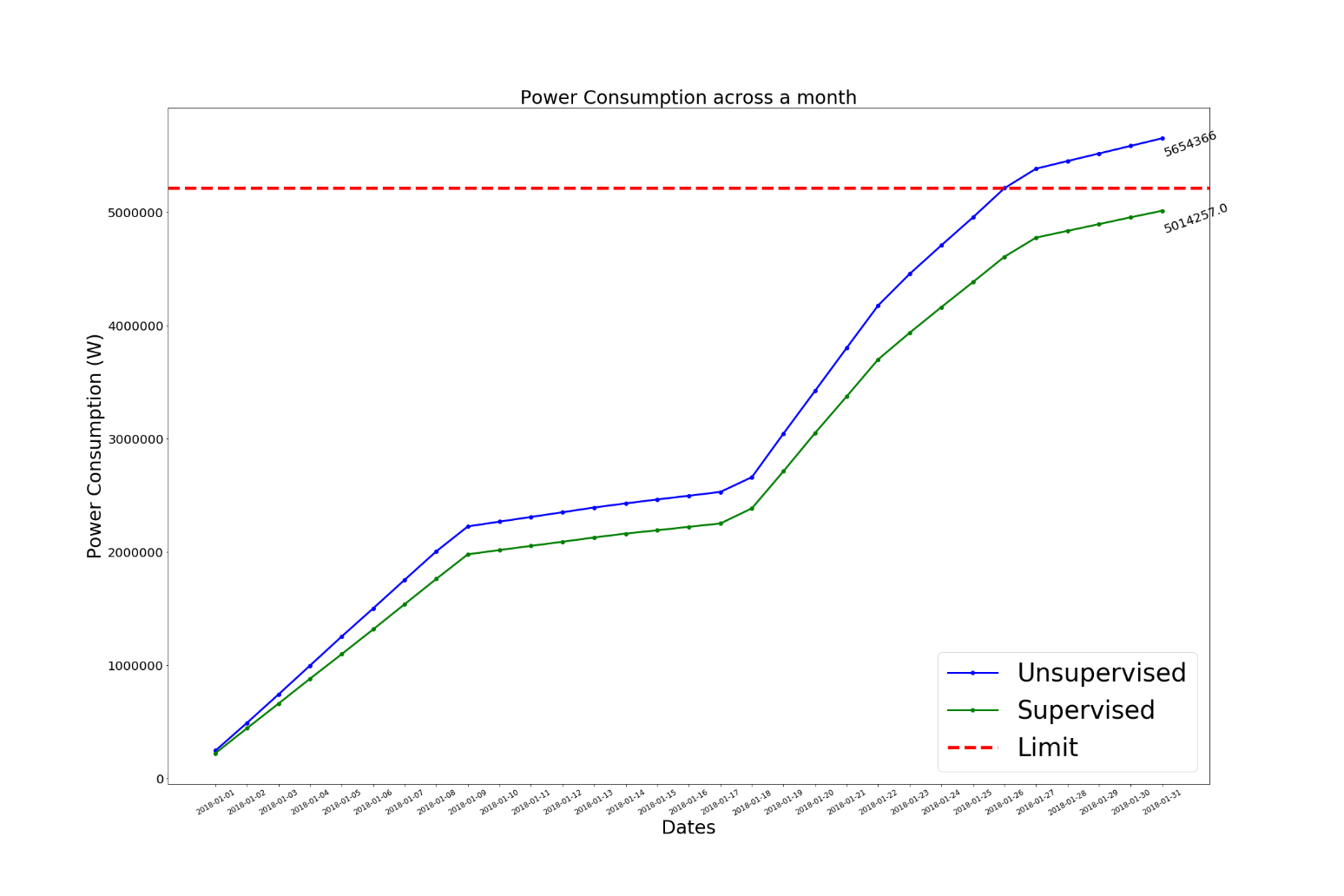
**4.2. Random Forest Regression**



***Figure 8 – Random Forest Regression***

**Figure 7** represents multiple linear regression, **Figure 8** represents random forest regression and **Figure 9** represents decision tree regression.

**4.3. Decision Tree Regression**



***Figure 9 – Decision Tree Regression***

All machine learning algorithms perform significantly well as shown in the graphs above. The green line represents the power consumption when there is a custom algorithm used. The blue line represents the power consumption when there is no algorithm used. As per the experiment, the green line falls below the red line (limit) hence indicates that it is supervised.

**4.4. Conclusion**

The performance data is collected based on the confusion matrix produced by the algorithms. In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm.

Kernel SVM and K Nearest Neighbours tabulated similar results. Logistic Regression and SVM showcased a significant improvement in the results. However, Decision Tree and Random Forest classifiers performed better than linear classifiers. Random Forest and Decision Tree both topped at 99.5% making them the best algorithm to use for this kind of dataset. Overall the results are similar in nature. We come to the conclusion that for a dataset of 1,000 rows, there are major differences in accuracy on machine learning models.

**References**

[1] Christian Beckel, Heinz Serfas, *Requirements for Smart Home Applications and Realization with WS4D-PipesBox*

[2] Guiqing Zhang, Xianghe Ji, Chengdong Li, Liang Tao, Xiaolong Wu, *Research on Energy-saving Control of Standby Household Appliances*

[3] Wellen S. Lima, Eduardo Souto, Thiago Rocha, *User Activity Recognition for Energy Saving in Smart Home Environment*

[4] Manoj Manivannan ID, Behzad Najafi and Fabio Rinaldi, *Machine Learning-Based Short-Term Prediction of Air-Conditioning Load through Smart Meter Analytics*

[5] Vibhatha Abeykoon, Nishadi Kankanamdurage, *Electrical Devices Identification through Power Consumption using Machine Learning Techniques*

[6] [E C Alexopoulos](https://www.ncbi.nlm.nih.gov/pubmed/?term=Alexopoulos%20EC%5BAuthor%5D&cauthor=true&cauthor_uid=21487487), Department of Public Health, Medical School, University of Patras, Rio Patras, Greece, *Introduction to Multivariate Regression Analysis*

**Appendix**

1. **Multiple Linear Regression**

# coding: utf-8

# # Applying Machine Learning and Deep Learning to identify home appliances consuming excess power

#

# ## Copyright (c) 2018, Faststream Technologies

# ## Author: Sudhanva Narayana

# In[1]:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

import pickle

from sklearn import metrics

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error

from sklearn.externals import joblib

# ### Import dataset ignoring headers

# In[2]:

df = pd.read\_csv('../data/trial\_1/home\_data.csv')

# ### Dataset

# In[3]:

df.head()

# ### Importing dataset

# In[4]:

X = df.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 8]].values

y = df.iloc[:, 9].values

# ### Encoding Categorical Variables

# In[5]:

# Encoding categorical data

labelencoder\_X\_0 = LabelEncoder()

X[:, 0] = labelencoder\_X\_0.fit\_transform(X[:, 0])

labelencoder\_X\_1 = LabelEncoder()

X[:, 1] = labelencoder\_X\_1.fit\_transform(X[:, 1])

labelencoder\_X\_2 = LabelEncoder()

X[:, 2] = labelencoder\_X\_2.fit\_transform(X[:, 2])

labelencoder\_X\_3 = LabelEncoder()

X[:, 3] = labelencoder\_X\_3.fit\_transform(X[:, 3])

labelencoder\_X\_4 = LabelEncoder()

X[:, 4] = labelencoder\_X\_4.fit\_transform(X[:, 4])

labelencoder\_X\_5 = LabelEncoder()

X[:, 5] = labelencoder\_X\_5.fit\_transform(X[:, 5])

labelencoder\_X\_6 = LabelEncoder()

X[:, 6] = labelencoder\_X\_6.fit\_transform(X[:, 6])

onehotencoder = OneHotEncoder(categorical\_features=[0, 1, 2, 3, 4, 5, 6])

hot\_X = onehotencoder.fit\_transform(X).toarray()

# ### Avoiding the dummy variable trap

# In[6]:

columns = df.columns

dummies = []

dummies\_sum = 0

categories = [0, 1, 2, 3, 4, 5, 6]

for category in categories:

dummies\_sum += category \* (df.iloc[:, category].unique().size)

dummies.append(dummies\_sum)

# Removing dummy variables

hot\_X = np.delete(hot\_X, dummies, 1)

# ### Splitting the dataset into the Training set and Test set (75%, 25%)

# In[7]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(hot\_X, y, test\_size=0.25, random\_state=0)

# ### Multiple Linear Regression

# In[8]:

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

# In[9]:

y\_pred = regressor.predict(X\_test)

y\_pred

# In[10]:

joblib.dump(regressor, '../model/multiple\_reg.pkl')

# In[11]:

print("Mean Absolute Error: ", metrics.mean\_absolute\_error(y\_test, y\_pred))

print("Mean Squared Error: ", metrics.mean\_squared\_error(y\_test, y\_pred))

print("Root Mean Squared Error: ", np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

1. **Random Forest Regression**

# coding: utf-8

# # Applying Machine Learning and Deep Learning to identify home appliances consuming excess power

#

# ## Copyright (c) 2018, Faststream Technologies

# ## Author: Sudhanva Narayana

# In[1]:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

import pickle

from sklearn import metrics

from sklearn.ensemble import RandomForestRegressor

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error

from sklearn.externals import joblib

# ### Import dataset ignoring headers

# In[2]:

df = pd.read\_csv('../data/trial\_1/home\_data.csv')

# ### Dataset

# In[3]:

df.head()

# ### Importing dataset

# In[4]:

X = df.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 8]].values

y = df.iloc[:, 9].values

# ### Encoding Categorical Variables

# In[5]:

# Encoding categorical data

labelencoder\_X\_0 = LabelEncoder()

X[:, 0] = labelencoder\_X\_0.fit\_transform(X[:, 0])

labelencoder\_X\_1 = LabelEncoder()

X[:, 1] = labelencoder\_X\_1.fit\_transform(X[:, 1])

labelencoder\_X\_2 = LabelEncoder()

X[:, 2] = labelencoder\_X\_2.fit\_transform(X[:, 2])

labelencoder\_X\_3 = LabelEncoder()

X[:, 3] = labelencoder\_X\_3.fit\_transform(X[:, 3])

labelencoder\_X\_4 = LabelEncoder()

X[:, 4] = labelencoder\_X\_4.fit\_transform(X[:, 4])

labelencoder\_X\_5 = LabelEncoder()

X[:, 5] = labelencoder\_X\_5.fit\_transform(X[:, 5])

labelencoder\_X\_6 = LabelEncoder()

X[:, 6] = labelencoder\_X\_6.fit\_transform(X[:, 6])

onehotencoder = OneHotEncoder(categorical\_features=[0, 1, 2, 3, 4, 5, 6])

hot\_X = onehotencoder.fit\_transform(X).toarray()

# ### Avoiding the dummy variable trap

# In[6]:

columns = df.columns

dummies = []

dummies\_sum = 0

categories = [0, 1, 2, 3, 4, 5, 6]

for category in categories:

dummies\_sum += category \* (df.iloc[:, category].unique().size)

dummies.append(dummies\_sum)

# Removing dummy variables

hot\_X = np.delete(hot\_X, dummies, 1)

# ### Splitting the dataset into the Training set and Test set (75%, 25%)

# In[7]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(hot\_X, y, test\_size=0.25, random\_state=0)

# ### Random Forest Regression

# In[8]:

regressor = RandomForestRegressor(n\_estimators=300, random\_state=0)

regressor.fit(X\_train, y\_train)

# In[9]:

y\_pred = regressor.predict(X\_test)

y\_pred

# In[10]:

joblib.dump(regressor, '../model/random\_forest.pkl')

# In[11]:

print("Mean Absolute Error: ", metrics.mean\_absolute\_error(y\_test, y\_pred))

print("Mean Squared Error: ", metrics.mean\_squared\_error(y\_test, y\_pred))

print("Root Mean Squared Error: ", np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

1. **Decision Tree Regression**

# coding: utf-8

# # Applying Machine Learning and Deep Learning to identify home appliances consuming excess power

#

# ## Copyright (c) 2018, Faststream Technologies

# ## Author: Sudhanva Narayana

# In[1]:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

import pickle

from sklearn import metrics

from sklearn.tree import DecisionTreeRegressor

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error

from sklearn.externals import joblib

# ### Import dataset ignoring headers

# In[2]:

df = pd.read\_csv('../data/trial\_1/home\_data.csv')

# ### Dataset

# In[3]:

df.head()

# ### Importing dataset

# In[4]:

X = df.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 8]].values

y = df.iloc[:, 9].values

# ### Encoding Categorical Variables

# In[5]:

# Encoding categorical data

labelencoder\_X\_0 = LabelEncoder()

X[:, 0] = labelencoder\_X\_0.fit\_transform(X[:, 0])

labelencoder\_X\_1 = LabelEncoder()

X[:, 1] = labelencoder\_X\_1.fit\_transform(X[:, 1])

labelencoder\_X\_2 = LabelEncoder()

X[:, 2] = labelencoder\_X\_2.fit\_transform(X[:, 2])

labelencoder\_X\_3 = LabelEncoder()

X[:, 3] = labelencoder\_X\_3.fit\_transform(X[:, 3])

labelencoder\_X\_4 = LabelEncoder()

X[:, 4] = labelencoder\_X\_4.fit\_transform(X[:, 4])

labelencoder\_X\_5 = LabelEncoder()

X[:, 5] = labelencoder\_X\_5.fit\_transform(X[:, 5])

labelencoder\_X\_6 = LabelEncoder()

X[:, 6] = labelencoder\_X\_6.fit\_transform(X[:, 6])

onehotencoder = OneHotEncoder(categorical\_features=[0, 1, 2, 3, 4, 5, 6])

hot\_X = onehotencoder.fit\_transform(X).toarray()

# ### Avoiding the dummy variable trap

# In[6]:

columns = df.columns

dummies = []

dummies\_sum = 0

categories = [0, 1, 2, 3, 4, 5, 6]

for category in categories:

dummies\_sum += category \* (df.iloc[:, category].unique().size)

dummies.append(dummies\_sum)

# Removing dummy variables

hot\_X = np.delete(hot\_X, dummies, 1)

# ### Splitting the dataset into the Training set and Test set (75%, 25%)

# In[7]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(hot\_X, y, test\_size=0.25, random\_state=0)

# ### Multiple Linear Regression

# In[8]:

regressor = DecisionTreeRegressor()

regressor.fit(X\_train, y\_train)

# In[9]:

y\_pred = regressor.predict(X\_test)

y\_pred

# In[10]:

joblib.dump(regressor, '../model/decision\_tree.pkl')

# In[11]:

print("Mean Absolute Error: ", metrics.mean\_absolute\_error(y\_test, y\_pred))

print("Mean Squared Error: ", metrics.mean\_squared\_error(y\_test, y\_pred))

print("Root Mean Squared Error: ", np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))